In the Specification

Please amend the specification of this application as follows:

Rewrite the paragraph at page 1, lines 14 to 15 as follows:

--U.S. Provisional Application No. ~~60/___,___ (TI-29494)~~ 60/224.607 entitled DATAPIPE ROUTING BRIDGE now U.S. Patent Application Serial No. 09/905,378; and--

Rewrite the paragraph at page 1, lines 16 to 18 as follows:

--U.S. Provisional Application No. ~~60/___,___ (TI-29497)~~ 60/224,913 entitled PULL TRANSFERS AND TRANSFER RECEIPT CONFIRMATION IN A DATAPIPE ROUTING BRIDGE now U.S. Patent Application Serial No. 09/905,379.--

Rewrite the paragraph at page 5, line 11 to page 6, line 7 as follows:

--This application uses the descriptive name datapipe routing bridge or simply datapipe to describe a packet based communications peripheral connecting multiple processors without glue logic or CPU intervention. Figure 1 illustrates the makeup of a datapipe. It is composed of three building blocks transmitter 101, bridge ~~102~~ 103 and receiver ~~103~~ 102. The main function of the bridge component is to provide high levels of connectivity between multiple digital signal processors without paying the penalties usually associated with inter-processor connections. Dedicated routing logic within the datapipe autonomously navigates data packets of programmable size along the shortest distance from the source processor to one or more destination processors. Transmitter 101 may transmit data packets via bridge 103 to one or both of the right and left ports. Transmitter 101 responds to transmit events and transmit interrupts from an associated data processor (not shown) to supply data from internal I/O memory 105 to bridge 103. Bridge 103 is capable of

retransmitting a data packet received at one of the right or left ports to the other port. Bridge 103 may also transfer a received data packet to receiver 102 in addition to or instead of retransmission at the other port. The actions of bridge 103 are determined by a header of the data packet. Upon receipt of a data packet, receiver stores the received data in internal I/O memory 105 and may generate a receive event to the associated data processor. In the preferred embodiment the associated data processor is a digital signal processor.--

Rewrite the paragraph at page 7, line 20 to page 8, line 6 as follows:

--Figure 3 illustrates the three components of the datapipe hardware at each terminal node and their connection to the datapipe network in an example data transfer. The transmit controller 301 drives the packets from internal I/O RAM 302 out ~~the pins~~ lines 303 to the links connecting the digital signal processors. The communications bridge 304 routes each packet into or around each digital signal processor node on the network. For each packet routed into a node from the network, the receive unit 305 pushes the packet into the local I/O RAM 306 of the destination digital signal processor. Both of the two external ports of the bridge feature two unidirectional channels, one for input and one for output. Both transmitter and receiver can send communications events to the interrupt selectors in the associated digital signal processor. The transmitter can also respond to interrupts from the interrupt selector. The receiver can also send an interrupt directly to the transmitter.--

Rewrite the paragraph at page 8, lines 14 to 28 as follows:

--Figure 4 illustrates the datapipe within a conventional digital signal processor integrated circuit. Internal I/O RAM

input buffers 405, when almost full, send an event to the chip direct memory access (DMA) unit to move the data into the level-2 (L2) main memory 401, where it can be accessed directly by the central processing unit core 400. Note that this application contemplates that central processing unit core 400 is a digital signal processor, however this invention is equally applicable to a general purpose data processor. Internal I/O RAM 405 of the datapipe is split into two independent blocks for simultaneous direct memory access unit and datapipe access. The direct memory access port servicing internal I/O RAM ~~406~~ 405 and the datapipe looks exactly like the other direct memory access ports driving the remaining chip peripherals. Figure 4 further illustrates conventional features of a digital signal processor including L2 instruction RAM 402, L2 data RAM 403, parameter RAM (PAR RAM), power down circuit (PWR DWN), phase locked loop circuit (PLL), first and second timers (TIMER0, TIMER1), a host port interface (HPI), two multi-channel buffered serial ports (McBSP0 and McBSP1) and a 32-bit external memory interface (EMIF32).--

Rewrite the paragraph at page 13, lines 10 to 18 as follows:

--The use of the BLOCK tx_opcode is similar to an indirect addressing mode using the same processor analogy. The data that the BLOCK tx_opcode transmits ~~is~~ has its address embedded inside the BLOCK tx_opcode, but the data itself is residing in a different area of memory. A BLOCK tx_opcode causes the transmitter to transfer a block of data from a different local I/O RAM location, whose address has been previously loaded into the transmitter address register with other tx_opcodes preceding the BLOCK tx_opcode.--

Rewrite the paragraph at page 14, lines 1 to 21 as follows:

--Figure 8 illustrates point-to-point packet routing protocol. A point-to-point packet is identified by a PTP rx_opcode 801 in its header. As the header enters the bridge component at a local node, the DST_NODE field 802 inside the PTP rx_opcode 801 is compared the 5-bit NODE_ADDR field of the bridge NODE_CFG Register 803. A successful address match 810 causes the packet to enter this local node through the bridge internal center port, across the receiver and into the active channel block of the local I/O RAM. A negative address match triggers the left port ID comparator 805 and right port ID comparator 806 that compare the decoded value of the DST_NODE field 802 against the two 32-bit resident direction registers, ID_RIGHT 812 and ID_LEFT 813. A successful right match at right port ID comparator 806 causes the packet to be routed out of bridge 103 through the right port 807 to another node in the network. A successful left match at left port ID comparator 805 causes the packet to be routed out of bridge 103 through left port ~~805~~ 808 to another node on the network. Left port ID comparator 805 and right port ID comparator 806 form a bitwise AND. A logical 1 in any bit location indicates a successful match.--

Rewrite the paragraph at page 14, line 22 to page 15, line 15 as follows:

--Figure 9 illustrates broadcast packet routing protocol. A broadcast packet is identified by a BCAST rx_opcode 901 in its header. As the header enters the bridge component at a local node, the 3-bit NAV field 902 inside the BCAST rx_opcode is evaluated to determine the port(s) through which the packet is going to leave the bridge. A value of logical 1 in the middle NAV bit causes the packet to enter this local node through the internal center port of the bridge, across the receiver and into the active channel block of the local I/O RAM. A value of logical 1 in the left bit of NAV

field 902 causes the packet to be routed out of the bridge through the left port 904 another node on the network. Similarly, a logical 1 in the ~~left~~ right bit of NAV field 902 causes the packet to be routed out of the bridge through the right port 905 another node on the network. Any combination of bits can be turned on inside NAV field 902, making it possible for the same data packet to both enter the node and be also routed out of the bridge through either left, right or both ports. Each BCAST rx_opcode is only used once per each intermediate node. After entering each node, the spent BCAST rx_opcodes are popped off the packet header and BCAST rx_opcode immediately behind it is used to navigate the packet through the next link on the datapipe network. As shown int Figure 9, the other bridge hardware is not used for broadcast packets.--

Rewrite the paragraph at page 15, lines 16 to 28 as follows:

--Figure 10 illustrates datapipe events, interrupts and configuration bits. Configuration of the datapipe is accomplished through a ~~27-bit~~ 26-bit CFG_BUS, which includes six inputs of reset and enable functions routed to the receiver, bridge, and transmitter, respectively. These are labeled 1001, 1002, and 1003 in Figure 10. A total of twenty-one monitor signals are routed back into the CFG_BUS 1000 I/O. These twenty one signals are: (a) two inputs from the transmitter labeled TX_STATE; and (b) seventeen event signals including TX_CIRC events (4), TX_STREAM events (2), RX_CIRC events (8), RX_STREAM events (3) and ~~©~~ (c) two interrupt signals ~~INT_UNEXP~~ INT UNXP 1006 and ~~INT_TOUT~~ INT RCPT 1007. The two interrupt signals ~~INT_UNEXP~~ INT UNXP 1006 and ~~INT_TOUT~~ INT RCPT 1007 are also monitored.--

Rewrite the paragraph at page 17, line 23 to page 18, line 9 as follows:

--The internal datapipe interrupt flag register delivers seventeen datapipe events to the chip interrupt selectors and receives two interrupts driving the datapipe transmitter and one interrupt driving the bridge. The INT_UNXP 1006 interrupt, if enabled, causes the transmitter to temporarily suspend batch transfers and to start processing the unexpected transfer script. The INT_RCPT 1007 interrupt, if enabled, causes the transmitter to temporarily suspend batch transfer and to start processing the transfer receipt script. ~~The INT_TOUT interrupt represents a timer time-out condition.~~ The eleven datapipe events are composed of eleven events from the receiver (eight circular events 1004 and three stream events 1005) and six transmitter events (four circular events and two stream events). All seventeen datapipe interrupt flags are mirrored by corresponding interrupt enable bits in the datapipe interrupt enable register. The seventeen datapipe interrupt flag have a persistence of one clock pulse period.--